

11/16/00  
10990 U.S. PTO

11.14.00

A

LAW OFFICES

LERNER, DAVID, LITTENBERG, KRUMHOLZ & MENTLIK, LLP

600 SOUTH AVENUE WEST

WESTFIELD, NEW JERSEY 07090-1497

SIDNEY DAVID  
JOSEPH S. LITTENBERG  
ARNOLD H. KRUMHOLZ  
WILLIAM L. MENTLIK  
JOHN R. NELSON  
ROY H. WEPNER  
STEPHEN B. GOLDMAN  
CHARLES P. KENNEDY  
PAUL H. KOCHANSKI  
MARCUS J. MILLET  
BRUCE H. SALES  
ARNOLD B. DOMPERI  
KEITH E. SILMAN  
ROBERT S. COHEN  
MICHAEL H. TESCHNER  
GREGORY S. GEWIRTZ  
JONATHAN A. DAVID  
SHAWN P. FOLEY\*

19081 654-5000  
FAX 19081 654-7866  
www.idlkm.com

PATENTS, TRADEMARKS AND COPYRIGHTS

November 16, 2000

THOMAS M. PALISI  
STEPHEN F. ROTH  
KIMBERLY V. PERRY  
RENEE M. ROBERSON  
LYNNE A. BORCHERS  
MICHAEL J. DOHERTY  
MATTHEW B. DERNER  
J. KIRKLAND DOUGLASS  
ROBERT J. SCHEFFEL  
SCOTT S. SERVILLA  
JEANNE P. VALLEBUONA  
EDWARD O. BERGAMANT  
FRANK J. CHESKY II

OF COUNSEL  
LAWRENCE J. LERNER  
DANIEL H. BOBIS  
RAYMOND W. AUGUSTINA  
HARVEY L. COHEN  
JEFFREY S. DICKEY

\*NORTH CAROLINA BAR ONLY  
\*NEW YORK BAR ONLY

BOX PATENT APPLICATION  
Assistant Commissioner For Patents  
Washington, D.C. 20231

File No.: INSTAK 3.0-001  
Title: METHOD AND SYSTEM OF DEPLOYING  
SERVER-BASED APPLICATIONS

Inventor(s): Pramod Khandekar  
Assignee: InstaKnow, Inc.

CUSTOMER NUMBER 000530

Dear Sir:

Enclosed herewith please find the following documents in the above-identified application for Letters Patent of the United States, which claims the benefit of Provisional Application Serial No. 60/174,747, filed January 4, 2000, Provisional Application No. 60/166,247, filed November 18, 1999, and Provisional Application No. 60/171,143 filed December 16, 1999.

- 1 Pages of Abstract  
41 Pages of Specification  
16 Number of Claims  
26 Sheets of Drawings ☐ A4 ☒ 11"  
X Applicant claims small entity status under 37 C.F.R. 1.27  
X Assignment and Recordation Form Cover Sheet

Declaration  
One (1) return-addressed postcard

PLEASE PROVIDE FILING DATE AND SERIAL NUMBER

Please charge Deposit Account No. 12-1095 in the amount of \$515.00 calculated as follows:

Basic Fee		\$	355.00
Additional Fees:			
Total number of claims (including multiple dependent claims):	16		
Total number of claims in excess of 20:	0 x \$9		0
Number of independent claims:	7		
Number of independent claims minus 3:	4 x \$40		160.00
Fee for multiple dependent claim(s) (\$135)			

TOTAL FILING FEE \$ 515.00

In the event the actual fee is greater than the payment authorized above, the Patent Office is authorized to charge any deficiency to our Deposit Account No. 12-1095.

Respectfully submitted,

LERNER, DAVID, LITTENBERG,  
KRUMHOLZ & MENTLIK, LLP

JONATHAN A. DAVID

Reg. No. 36,494

382608\_1

EXPRESS MAIL LABEL NUMBER:

EL479162096US

Applicant or Patentee: Pramod Khandekar  
 Application or Patent No.:  
 Filed or Issued:

Attorney's  
 Docket No.  
 INSTAK 3.0-001

Title: METHOD AND SYSTEM OF DEPLOYING SERVER-BASED APPLICATIONS

STATEMENT CLAIMING SMALL ENTITY STATUS  
 (37 C.F.R. §§ 1.9(f) and 1.27(c)) - SMALL BUSINESS CONCERN

I hereby state that I am

- ☐ the owner of the small business concern identified below:  
☒ an official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF SMALL BUSINESS CONCERN: InstaKnow.com Inc.  
 ADDRESS OF SMALL BUSINESS CONCERN: 40 Brunswick Avenue  
 Edison, New Jersey 08817

I hereby state that the above-identified small business concern qualifies as a small business concern as defined in 13 C.F.R. Part 121 for purposes of paying reduced fees to the United States Patent and Trademark Office. Questions related to size standards for a small business concern may be directed to: Small Business Administration, Size Standards Staff, 409 Third Street, SW, Washington, DC 20416.

I hereby state that rights under contract or law have been conveyed to and remain with the small business concern identified above with regard to the invention described in

- ☒ the specification filed herewith with title as listed above.  
☐ the application identified above.  
☐ the patent identified above.

If the rights held by the above-identified small business concern are not exclusive, each individual, concern or organization having rights in the invention must file separate statements as to their status as small entities, and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 37 C.F.R. § 1.9(c) if that person made the invention, or by any concern which would not qualify as a small business concern under 37 C.F.R. § 1.9(d), or a nonprofit organization under 37 C.F.R. § 1.9(e).

Each person, concern, or organization having any rights in the invention is listed below:

- ☒ no such person, concern, or organization exists.  
☐ each such person, concern, or organization is listed below.

\*NOTE: Separate statements are required from each named person, concern or organization having rights to the invention stating their status as small entities. (37 C.F.R. § 1.27)

FULL NAME

ADDRESS:

☐ INDIVIDUAL ☐ SMALL BUSINESS CONCERN ☐ NONPROFIT ORGANIZATION

FULL NAME:

ADDRESS:

☐ INDIVIDUAL ☐ SMALL BUSINESS CONCERN ☐ NONPROFIT ORGANIZATION

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate (37 C.F.R. § 1.28(b)).

NAME OF PERSON SIGNING: Pramod Khandekar

TITLE OF PERSON IF OTHER THAN OWNER: President

ADDRESS OF PERSON SIGNING: 40 Brunswick Avenue, Edison, New Jersey 08817

P. S. Khandekar  
 SIGNATURE

11/15/2000  
 DATE

METHOD AND SYSTEM OF DEPLOYING SERVER-BASED APPLICATIONS  
CROSS-REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of U.S.  
5 Provisional Application No. 60/174,747, filed January 4,  
2000, U.S. Provisional Application No. 60/166,247, filed  
November 18, 1999, and U.S. Provisional Application  
No. 60/171,143, filed December 16, 1999, the disclosures  
of which are hereby incorporated by reference herein.

10 FIELD OF THE INVENTION

The present invention relates to business rules  
based application development and the deployment of such  
applications. In particular, the present invention  
provides for the development of customized applications  
15 using basic business rules and logic such that a software  
programmer is not needed and provides for the deployment  
of such applications on server computers in response to  
client browser requests without the need for business  
logic coding in multiple places on the server.

20 BACKGROUND OF THE INVENTION

The Internet and World Wide Web (the "Web") are  
expanding globally with millions of new users being added  
every month. This expansion has resulted in more and  
more business processes being deployed on Web servers.  
25 Web servers are Web-connected computers that receive  
requests from client Web browsers, run the required  
application processes, and send the response back to the  
client Web browser for the next action from the client.

A business process is a unique sequence of detailed  
30 business actions carried out at a specific time to

achieve a specific business result. Each process has a distinct start and a distinct end point. A typical application has many processes that can be run independent of each other.

5 Conventionally, to develop customized software applications, a business user either had to be a computer programmer or hire a computer programmer to write code to implement the desired business logic. In the case of hiring a programmer, the business user typically has to  
10 wait for weeks or months in order to get the application built, tested, debugged and operational.

Likewise, to deploy applications on a Web server, program code is written and deployed on the Web server. Depending upon inputs from the client, this code executes  
15 a specific set of business logic, which is typically kept in a special program called a DLL (Dynamic Link Library) and generates HTML (Hypertext Markup Language) output to be sent to the client's Web browser.

A DLL is a set of routines that can be called from  
20 procedures and is loaded and linked into an application at run time. HTML is language used to create documents on the Web with hypertext links. HTML defines the structure and layout of a Web page by using a variety of tags and attributes. An HTML tag consists of a  
25 directive, possibly extended with one or more attributes, within angle brackets, for example <FONT SIZE=3>. There are many such tags that can be used to format and layout the information on a Web page. For instance, the tag <P> is used to make paragraphs and <I> ... </I> is used to  
30 italicize fonts. Tags can also specify hypertext links,

which automatically direct users to other Web pages with a single click of the mouse on the link.

On Microsoft operating systems, a Microsoft supplied component known as ASP (Active Server Pages) is required  
5 to get the request from the Web browser, run the appropriate application logic or DLL, construct an HTML output, and then send the output to the browser. ASP allows Web pages to be dynamically created by the Web server and uses scripting known as ActiveX, which  
10 provides a set of rules for how applications share information.

ActiveX uses COM (Component Object Model) components, which are binary files (such as .DLL, .ocx, or .exe files) that support the Microsoft COM standard  
15 for providing objects. Objects are generally entities that consists of both data and instructions for how to manipulate the data. COM components enable programmers to develop objects that can be accessed by any COM-compliant application. ActiveX and ActiveX controls are  
20 based on COM. ActiveX controls can be developed using a variety of programming languages such as C, C++, Java and Visual Basic. An ActiveX control, for example, can be automatically downloaded and run by a Web browser.

With ASP, one can combine HTML pages, script  
25 commands, and COM components to create interactive Web pages or Web-based applications. When a Web browser requests a Web page created by ASP (i.e., a Web page with a .ASP file extension), the Web server computer generates a page with HTML code and sends it back to the Web  
30 browser.

A convention use of ASP to deploy Web server applications is shown in FIG. 1, where a client browser computer 10 communicates via the Internet or Web 11 with Web server computer 12. Web server computer 12 includes the typical components found in a Web server computer, including for example, ROM and RAM memory, hard drive memory, a microprocessor, monitor, keyboard, mouse, etc. Web server 12 computer is configured with ASP software, which includes an ASP router component 14, DLL components 16, an HTML builder component 18, and a final HTML output component 19. DLL component 16, in this example, consists of three discrete processes including a Customer DLL 16a, an Order DLL 16b and a Shipment DLL 16c. Associated with each of ASP router component 14, DLL component 16, and HTML builder component 18 is separate business logic, which typically is programmed for each such component by a computer programmer familiar with ASP (ASP scripting. Thus, the use of ASP requires that the business logic be spread over many different components of ASP (ASP router, DLL business Logic, HTML builder and HTML output components) and requires specific programming skills and knowledge.

#### SUMMARY OF THE INVENTION

One aspect of the present invention provides a computer-implemented method for running applications on a server computer and generating Web page information to be displayed on one or more client computers connected to the server computer via the Internet. The method comprises: sending data from a client computer to a server computer running a DLL, the data comprising an identify of a user selected application, Web page

information associated with a Web page displayed on the client computer, and user-entered information used with the application; running the DLL to retrieve the data and to identify one or more executable processes within the selected application; executing the identified processes within the DLL in association with the user-information to generate output information; generating Web page information used to form a Web page viewable at the client computer by the DLL, the Web page information containing the output information; and forwarding the Web page information to the client computer for display.

Preferably, the Web page information is forwarded to an ASP layer and comprises HTML output, and preferably HTML tags. The Web page information preferably comprises an incoming screen name from the current screen being viewed at the client computer. The DLL also desirably runs a routing subroutine to select one or more executable processes by reference to parameters of incoming screen name, application file name and a selected screen element. Most preferably, the client computer is configured to run a Web browser for sending and receiving information to and from the server computer.

Another aspect of the present invention provides a computer-implemented method for running applications on a server computer connected via a network to one or more client computers, comprising: receiving requests at the server computer from a client computer running a browser program; running an application in response to the requests; constructing coded information used to form output by the browser program based on results generated

from the application; and forwarding the coded information to the browser program, wherein all necessary business logic for receiving the requests, running the application, and constructing the coded information is contained within a single linkable library of executable functions. Preferably, the single linkable library of executable functions resides exclusively in the Web server computer and comprises a DLL.

A further aspect of the present invention provides a computer-implemented method for running applications in a networked computing environment. This method comprises: sending requests from a client computer running a browser program; receiving the requests at a server computer; running an application in response to the requests; constructing output information at the server computer in a format used by the browser program based on results generated from the application; and forwarding the output information to the browser program for display at the client computer, wherein all necessary business logic for receiving the requests, running the application, and constructing the output information is contained within a single linkable library of executable functions.

Another aspect of the present invention provides a computer-based system for running applications in a networked computing environment. The system comprises a client computer running a browser program and a server computer for receiving requests from the client computer via network connecting the client and server computers. The server computer runs an application to respond to the requests, construct output information in a format usable by the browser program based on results generated from



the application, and forward the output information to the browser program for display at the client computer. All of the necessary business logic for receiving the requests, running the application, and constructing the  
5 output information is contained within a single linkable library of executable functions.

A still further aspect of the present invention is a computer readable medium storing a set of instructions for controlling a server computer in a networked  
10 computing environment including a client computer running a browser program, a server computer for receiving requests from the client computer, and a computer network interfacing the client and server computers. The medium comprises a single linkable library of executable  
15 functions and a set of instructions resident in server computer for causing the server computer to (i) run an application to respond to requests from the client computer, (ii) construct output information in a format used by the browser program based on results generated  
20 from the application, and (iii) forward the output information to the browser program for display at the client computer, wherein all necessary business logic for receiving the requests, running the application, and constructing the output information is contained within  
25 the single linkable library of executable functions. Preferably, the application is run at least in part on server computer.

Another embodiment of the present invention provides a computer-implemented method for developing a business  
30 rules based application, comprising: providing a set of wizards selectable by a user; providing a set of verbs

selectable by the user to implement business rules used in the application; choosing at least one of the verbs to define an action to be performed by the application when executed; and selecting at least one of the wizards to specify actions to be performed by the application when executed.

Desirably, the computer-implemented method for developing a business rules based application, comprises providing a set of wizards selectable by a user, including a data extraction wizard and an assignment wizard; providing a set of verbs to be used to implement business rules used in the application; choosing a verb to define at least one input criteria to be specified by the user of the application when executed; selecting the data extraction wizard to instruct the application to extract initial data from a first location; selecting the assignment wizard to specify a location for the application to store the initial data; selecting the data extraction wizard to instruct the application to extract secondary data from a second location, the secondary data being selected based on the initial data; and selecting the assignment wizard to specify a location for the application to store the secondary data.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram depicting the logical structure of a prior art approach to deploying Web server applications using ASP.

FIG. 2 is a block diagram showing the interconnection of various computers used in accordance with a preferred embodiment of the present invention.

FIG. 3 is a block diagram depicting the logical structure of the deployment of Web server applications in accordance with a preferred embodiment of the present invention.

5        FIG. 4 is depiction of a default Web page showing otherwise hidden controls associated with such Web page.

FIG. 5 is depiction of a select application Web page showing otherwise hidden controls associated with such Web page.

10       FIG. 6 is depiction of a user input Web page showing otherwise hidden controls associated with such Web page.

FIG. 7 is depiction of a results Web page showing otherwise hidden controls associated with such Web page.

15       FIG. 8 is a depiction of a main screen of a user-interface of a program implementing the present invention.

FIG. 9 is a depiction of a screen in the Application Wizard of the program.

20       FIG. 10 is a depiction of a screen in the Process Wizard of the program.

FIG. 11 is a depiction of a screen used to define a verb to be used with the program.

FIG. 12 is a depiction of a screen in the Data Extraction Wizard of the program.

25       FIG. 13 is a depiction of a screen in the Get Relational Data Wizard of the program.

FIG. 14 is a depiction of another screen in the Get Relational Data Wizard of the program.

FIG. 15 is a depiction of a further screen in the Get Relational Data Wizard of the program.

FIG. 16 is another screen in the get Relational Data Wizard of the program.

5        FIG. 17 is another screen in the Get Relational Data Wizard of the program.

FIG. 18 is a depiction of a screen used in the Data Extraction Wizard of the program.

10       FIG. 19 is a depiction of a screen in the Position Wizard of the program.

FIG. 20 is a depiction of a screen in the Assignment Wizard of the program.

FIG. 21 is a depiction of a screen in the Loops Wizard of the program.

15       FIG. 22 is a depiction of a screen of the Condition Builder Wizard of the program.

FIG. 23 is a depiction of a screen in the Assignment Wizard of the program.

20       FIG. 24 is a depiction of a screen in the Data Extraction Wizard of the program.

FIG. 25 is a depiction of a screen in the Assignment Wizard of the program.

FIG. 26 is a depiction of a screen in the Position Wizard of the program.

25       FIG. 27 is a depiction of a screen in the Assignment Wizard of the program.

FIG. 28 is a depiction of a screen in the Position Wizard of the program.

FIG. 29 is a depiction of a screen in the Save As Wizard of the program.

FIG. 30 is a depiction of a screen in the Condition Wizard of the program.

5        FIG. 31 is a depiction of a screen in the Display Wizard of the program.

FIG. 32 is a depiction of another screen in the Display Wizard of the program.

10       FIG. 33 is a depiction of the final data output generated by the program.

FIG. 34 is a depiction of a screen from the main menu of the program.

FIG. 35 is a depiction of the Scheduled Designer of the program.

15       FIG. 36 is a depiction of the Schedule Wizard of the program.

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

The present invention is preferably implemented in the form of software that is adapted to run on a Web server computer. The software can be stored on storage media such floppy disks, CD-ROM, hard disk, RAM, etc. and installed on the Web server. In a typical Web server set-up, as shown in FIG. 2, a Web server 24 is adapted to connect to the Internet or the Web 24 in the typical way to deliver Web pages to client computers 26. Client computers 26 run software such as Web browsers and connect to the Web in the typical fashion (e.g., dial-up access, cable modem, DSL, T-1 connection, etc.). Typical Web browsers include Microsoft Internet Explorer or

20

25

Netscape Navigator, running on operating systems such as Microsoft Windows (CE, 95, 98, NT, 2000), Mac OS, DOS, Unix, etc. Client computers 26 can comprise other devices beyond PCs, which connect to the Web in a wired or wireless fashion such as PDAs, notebook computers, mobile phones, etc. Web server 20 typically has a unique IP (Internet Protocol) address and associated domain name. Entering a URL (Uniform Resource Locator) into a Web browser running on a client computer 26 sends a request via the Web 24 to Web server 20, which then acts on the request, and fetches and sends a Web page back to the client's browser program for display. Web server 20 can comprise one or more computers (e.g., PC, Macintosh, mainframe, mini-computer, etc.) and runs the necessary underlying software to allow it to be connected to the Web and communicate with Web browser.

Turning to FIG. 3, a block diagram of the logic structure of elements or modules used in accordance with one aspect of the present invention is shown. Client browser 30 is connected to Web server 32 via the Web 31 in the conventional manner (dial-up, T-1, cable modem etc.). Web server 32 runs software, which provides a "blind" ASP-type layer without business logic, which, in accordance with the present invention, is now provided within single DLL 36. DLL 36 includes a router component 38, business component 40, HTML output component 42, and HTML forwarding component 44. Business logic for the routing component 38, application component 40 and HTML output component 42 is stored within the single DLL 34 instead of being stored in each associated component. Business component 40, in this example, includes separate processes "Respond To HomePage" process 40a, "Respond To

Select Application" process 40b, and "Respond To Customer Details" process 40c, discussed in further detail below. The software outputs the final HTML output to the client browser 30 at final HTML output component 46.

5 As explained below, another aspect of the present invention allows a user with little to no programming experience to define an application, which is a collection of business processes related to each other, to achieve a complete business objective. The output of  
10 the user's actions is an "application file" such as application file 40, containing all information required to execute the application. Using the deployment method of the present invention, part of the application file, along with some other required information, is passed to  
15 the DLL. The DLL then executes the application and necessary processes and returns back with the application result.

For example, the present invention could run a "Shipment" application that has a number of business  
20 processes that can run independently of each other, including:

1. Receive a shipment order
2. Check for availability of requested items
3. Confirm shipment mode

25 ... etc.

In accordance with a preferred embodiment, the first page shown to the user's browser is an HTML page that shows a welcome message. This page has within it two hidden controls (fields or data areas). One hidden  
30 control stores the path of the application file to be executed and the other hidden control stores the screen

name that uniquely identifies the present screen. For the default page, the values for these hidden controls are hard coded.

5       The welcome page does not have any input controls and shows a "Continue" button. When the user clicks on the Continue button, a call is made to the DLL. The DLL uses values of the hidden controls, i.e., the path of the application file and screen name, to determine from which application and screen the message is coming from and  
10       identifies the correct application file and process within that file to use to handle the message. The DLL applies the business logic as dictated by the process, composes an HTML reply to be sent to the client's browser, populates the hidden controls in the new HTML  
15       output and forwards it to the ASP layer. The HTML output is sent back as response to the user's request by the ASP layer. In this case, the ASP layer acts only as a blind pass-through mechanism, with no knowledge about what the HTML being sent to the user contains.

20       The next page that is shown to the user could contain some input parameters requiring the user to fill in some values. This page will also contain the same hidden controls as the default page, only this time these controls will not be hard coded, but will contain values  
25       populated by the DLL. After the user fills in values into the input controls and clicks on Continue button, a call is made to the DLL. Information contained in the hidden controls as well as any inputs provided by the user, is sent to the DLL. The DLL executes the application  
30       specified by the application path and generates an HTML output. Again, the HTML output is sent back to the user



as response. This process of showing the user pages for some data entry, calling the DLL with the user entered information (and hidden controls), and returning back with the HTML generated by the DLL is repeated until all  
5 inputs required to show the final result are received by the DLL. At that time, the HTML generated by the DLL is the final result to show to the client.

In the entire process, there is no need of writing any business logic or application specific logic into the  
10 Web server side code (ASP). All of the business logic and application specific logic resides within the DLL. Writing complicated Web server side code is now reduced to a simple call to the DLL, irrespective of the application being executed. Applications can hence be  
15 deployed much faster on the Web using the present invention.

As another example, a business user may want to build an application that shows a list of three applications to the user. The first is Customer Data  
20 Entry, the second is Order Details, and the third is Shipment Details. Each application has a different set of input requirements. As shown in FIG. 4, the first HTML page (the default page) that the user sees is the Welcome page 40. After that, as shown in FIG.5, the user sees an  
25 HTML page 50 showing a list of applications 52a, 52b, and 52c to choose from. When the user selects any one application, on the next page 60, shown in FIG. 6, the user is prompted for inputs 62a, 62b, 62c, and 62d specific to the application chosen to run. After entering  
30 the required inputs, the application is executed and the

user is shown the result on Web page 70, as shown in FIG. 7.

The first step is to define the application files for specified applications of Customer Data Entry 52a, 5 Order Details 52b and Shipment Details 52c. This is done preferably using the other aspect of the present invention; namely the methodology for building applications using a programmer-less, point-and-click, business rules based system. Preferred software for 10 implementing this aspect of the present invention is the design time version of InstaKnow™ software, offered by InstaKnow, Inc. of Edison, New Jersey. An example of the use of the business rules based application development aspect of the present invention is described below. Use 15 of this aspect of the invention builds an application file, which generates the screen showing the list of applications to show to the user.

On the default HTML page (FIG. 4), the application file path hidden control 42 is initialized to point to 20 the application file that generates the screen showing the list of applications to show to the user. Here, that application file is found at the address "C:\INSTAKNOW\INSTAWEB." The other hidden control, screen name 44, is given a unique screen name that is 25 understood by the DLL. Here, that name is "HOMEPAGE." The DLL has business logic within it to perform various functions depending on the incoming screen name.

When the user logs on to the Web site, he or she is shown the default HTML page 40, and hidden controls 42 30 and 44 are hard coded on this page. When the user clicks on the Continue link 46, a call is made to the DLL

resident on the Web server computer. At this stage, there is no input information being entered by the user on welcome page 40 so only the information contained in the hidden controls is passed to the DLL.

5       The DLL then executes the application identified by the application file path, performs any business logic depending on the incoming screen name, and generates an HTML output to send back to the user. Psuedo-code specifying this process is as follows:

```
10           If ScreenName = "HomePage"
              Call Process RespondToHomePage
          Else If ScreenName = "SelectApplication"
              Call Process RespondToSelectApplication
          Else If ScreenName = "CustomerDetails"
15           Call Process RespondToCustomerDetails
              ...
              ...
          End If
```

20       Psuedo-code for processing the RespondToHomePage application, is as follows:

```
          Process RespondToHomePage
              Show appropriate controls on next page
              Populate hidden controls
              Generate HTML for next screen
25       End Process
```

For the example, the HTML generated by the DLL will show the user the list of applications to choose from, as shown in FIG. 5. This HTML will also make sure that the hidden controls 54 and 56 are populated with appropriate values.

30

Suppose that the user selects Customer Data Entry 52a and clicks on Continue 58. The DLL is called again. This time, information contained in the hidden controls 54 and 56, as well as user entered information (selection  
5 of Customer Data Entry application) is passed to the DLL. The DLL executes the application and generates the HTML output to show to the user a page 60 (FIG. 6), asking for specific information for Customer Data Entry.

The user is then shown page 60 requesting input  
10 specific to the Customer Data Entry. For this example, the user enters information in the fields Name 62a, Date of Birth 62b, City 62c, and E-mail address 62d. When the user clicks on Continue 63, the DLL is again called with information entered by the user and the hidden controls  
15 64 and 65. The DLL executes the application and returns back with the result Web page 70 (FIG. 7), which also contains hidden controls 74 and 76.

With this aspect of the present invention, all of the business logic, including the logic required to get  
20 the request from the Web browser, run the appropriate application logic, and construct an HTML output is kept inside the DLL only. The ASP layer is blind and therefore used only to forward HTML generated by the DLL to the Web browser at process 44. This approach is  
25 possible because the present invention has the ability to (1) route messages from the Web browser to a correct process within the DLL and (2) generate the HTML output. In effect, the present invention is an alternative to ASP in both regards. To route to the correct process within  
30 the DLL, three hidden parameters are obtained from the HTML output generated by the DLL. These parameters

include (1) the screen name, (2) the application file (IAL) name, and (3) the clicked element, which is on the HTML page. With the combination of these three parameters, the program checks the router process and  
5 routes the logic control to the appropriate process within the DLL.

The above aspect of the present invention results in the benefit in that one does not need to learn ASP to deploy Web based applications. Another aspect of the  
10 present invention provides the business logic portion to be implemented using a user-friendly, wizard-based point and click programming tool interface, such as via the use of the software program InstaKnow<sup>SM</sup>, such that the business logic required for the routing, processing and  
15 HTML generation of any application can be specified by a business user without needing a computer programmer to write the code.

This application development aspect of the invention allows a business user to make customized Web and non-Web  
20 applications using a point-and-click protocol and without needing a programmer or knowledge of any programming language. All the user needs to know is the business data and its business meaning and the business processes (i.e., a particular sequence of operations on the data)  
25 that operate on that data to achieve the required business result.

In today's business world, the advantages of programmer-less, point-and-click, user driven application generation are obvious. With the present invention,  
30 users do not have to wait for weeks and months to get applications built, and they can build them themselves.

The invention also has a unique ability of "automatic surfing," i.e., automatically, enter data on Web pages, press buttons and links, get the response from the Web site and read data of interest to take further automated  
5 actions.

The invention is preferably in the form of software providing a point-and-click approach and wizard-based interface and assumes and requires no programming expertise among business users. The user just has to  
10 know in simple ways what manipulations have to be applied to the business data to achieve the required results. A user interface guides the user from that point on using a set of intelligent wizards. The wizards make sure that the complete and correct information about the user's  
15 intent for every step is correctly captured and saved. The applications built can be tried out and tested immediately without having to know technical details. Because of the automated assistance provided by the wizards, it is estimated that the users can build  
20 applications in much less time than it takes programmers of conventional languages to build the same applications.

The application development software of the present invention is not specific to any industry or computing problem. It can be used to easily develop simple or  
25 complex applications in any industry. As an example, a user can easily develop an application within a couple of days that will go to Web sites of its various supply vendors, collect spare part specific information, perform logical and mathematical operations on the data, decide  
30 which vendor gets the contract, inform the vendor, and

place an order for a certain quantity of spare parts over the Internet.

The software allows users to access and operate on real time data elements from various data sources. Data  
5 can be extracted from the Web (Internet and Intranets) from HTML or XML pages, and ODBC compliant data sources like fixed format flat files, delimited flat files, Excel, Word, Access, SQL Server, Oracle, Sybase and any other ODBC compliant databases. Another feature of the  
10 application development software is the ability to transfer not only data but also business logic over the Internet between collaborating computers.

Preferably, the application development software has built-in facilities for version control,  
15 security/permissions control and migration control, and can run on any 32-bit Microsoft platform on a client desktop or a LAN/Web server. In the Web server mode, high-end scalability can be ensured by deploying it as a business object component in Microsoft Transaction Server  
20 (MTS). A built-in scheduler is also preferred to allow repeated automatic executions of same business processes at user specified frequencies.

There are three main phases of the user's interaction with the application development software's  
25 point and click environment. These include a Design Phase, a Playback (Debug) Phase and a Run Phase. The Design Phase is where the user defines the business logic. The Playback (Debug) Phase is where the user can try out the business logic immediately. The user may  
30 start/stop the business logic after each step for verification (debugging) purposes. A watch mode allows

monitoring data values of interest as the business process is being verified. At the Run Phase, after being satisfied that the logic is working correctly, the user can schedule the business logic to run automatically at pre-determined frequencies, including on demand. In the run phase, the logic runs automatically without any user intervention.

An example of the use of the application development software is now described in the context of obtaining data from the Web and saving it into a spreadsheet. In this example, there is a list of stocks in a Microsoft Excel spreadsheet and the user wishes to retrieve and save the latest stock price and trade volume information for the whole list from the Web and then save it in another Excel spreadsheet. To achieve this business result, the following data manipulation steps have to be applied:

- Read list of users stocks from input spreadsheet
- Go to the MSN Investor Web page
- Start from top of the list
- Get the current item from the list, and loop until last stock in the list
  - Enter the stock symbol from the list into the stock symbol on the MSN Investor Web page. Press Get Quote button on Web page.
  - When the Web server responds, get the current stock price, volume, etc. from Web page



- Copy this information to the same numbered item in another (output) list
- Get the next item in input list
- 5      ➤ Repeat the loop
- Save the output list in another spreadsheet

Within the wizard-based-programming environment, the information about all the groups, data elements, and  
10 processes is always available once they have been defined. This information is arranged in a fashion that will facilitate the meaningful information exchange between the individually defined pieces, and produce a desired result.

15      The following describes how an application is built from the various components like data elements, groups, processes, and applications. The design phase is where the user defines his intent of what should occur to the business data of interest. The user selects the next  
20 action to be performed from a list of available actions.

In the example of getting stock prices and saving them in another spreadsheet, the actions performed are:

- Accept user input
- Read
- 25      • Go to Web site
- Start from top
- Loop until no more items in list
- Enter on Web site
- Get information from Web page

- Copy data
- Get next item in list
- Repeat loop
- Save
- 5 • Check if asked to display the result
- Display the result (if asked to)

The wizard-based programming interface guides the user from that point on as a set of intelligent wizards. The wizards make sure that the complete and correct information about the user's intent for every step is correctly captured and saved. As shown in FIG. 8, a user specified "process tree" 80 is shown to the user for easy visual identification of what steps have been specified so far.

15 The first step in the design of the application is to assign a name to the application. To do so, the designer clicks the Application button 82 to bring up the Application Wizard. The Application Name 92 is entered in the Wizard, and the stage of the application is specified at drop down box 94.

20 The next step is to define a process by which the application will be identified, using Process Wizard 100 (FIG. 10). The Name 102 for this process is as it appears on the screen in the design environment. The stage 104 of the application also is identified as before. If the designer is about to build a new process, Trial stage can be selected so that the designer can make changes during the process configuration as many times as needed.

The next field on the Process Wizard 100 is the permissions 106 to use this application. The read/write/execute permission will allow any other user to edit the design of this process. For tighter  
5 security, the designer can opt for a Read/Execute type of permission only. The next fields are the color scheme 108 and the language field 109. The designer can select the language in which the application will be built up. In this example the designer selected English as the  
10 language for application construction.

The first step under the process steps in the design of the above application is to ask the user if he/she wants to view the final output or not. This is achieved through the Accept verb. The web based programming such  
15 as InstaKnow, accomplishes complex programming tasks transparent to the user, by allowing the user to define a simple business action. This way, the user is least bothered with how to program the statement; while at the same time can construct logic in plain simple way.  
20 Intelligent wizards prompt the user to get complete and correct information about the user's business intent. The supported actions or verbs are specified for selection by the designer. For example, if the process, when it runs, needs to get some business information from  
25 the end user, the application designer uses the Accept verb to halt the program and prompt the end user for appropriate information. The user-supplied value, after designer specified validations, is stored in the data element specified by the designer. Preferred actions or  
30 verbs are those used by the InstaKnow software, which are listed under the Program Logic heading in U.S.

Provisional Application 60/174,747, the disclosure of which is hereby incorporated by reference herein.

As shown in FIG. 11, the designer has defined a local variable called Display Flag 110 local to this process (P001\_DisplayFlag). The designer gives a user-friendly name Accept Display Flag 116 to this accept step. Thereafter, the designer has to select a group from drop down box 111. The data element that the designer has defined does not belong to any of the groups. Also, it will be used within this process only, and occurs only one time in this process. Hence, the designer had categorized this data element as Local Single Instance. The designer selects "LocalSingleInstance" from the drop down box 111 of the "Select the Group." This fills up the grid below it with all the data elements that have been defined earlier as Local Single Instance. The designer highlights the desired data variable 110 and adds it to the bottom most grids using the Add One button 112. In this example, the data element, P001\_DisplayFlag 113 is chosen from the upper grid and displayed in the lower grid. The designer has to provide a message 114 that will be used to prompt to the user of this application. The designer wants to ask the user here whether he/she is interested in viewing the final output. Therefore he/she puts prompt 114 as "Do you wish to see the output display? (Yes/No)." This completes the design of an Accept verb. The designer clicks the OK or check button 115, and the Accept statement is added to the application in the designer environment.

In step 2, the designer wants to extract non-Web data by opening an existing Excel file to read the list

of stock symbols it contains. The read of Excel file is an Extract Data type of operation, which falls under the Program mode. The user clicks the Program mode to access the Extract verb. The click on extract button brings up  
5 the Extract Data Wizard 120 (shown in FIG. 12).

The designer must then define the data source. To do so, the designer supplies a name 121 to this extract procedure, and user-friendly comments 122. Thereafter the designer decides from where the data will be  
10 extracted. In this case, the data is to be extracted from an Excel file, which falls under the Other Data Sources category 123. A click on this button brings up the next Wizard 130 (FIG. 13) that will assist in extracting information from Excel files.

15 The designer provides a group name 131 by which the extracted list of stock symbols will be recognized, then selects the data type 132, which is Microsoft Excel, and the path 133 to the location of the file, which on the C drive.

20 The file is opened and information in it is read, and saved as database. The databases read are presented for selection in the next wizard step 140 (FIG. 14). In this example, the data is present in an Excel worksheet called Sheet 1. Hence, the information from this Excel  
25 workbook is read as Sheet1\$Database 142.

In the next wizard step, shown in FIG. 15, the user is shown the columns listed in the Selected spreadsheet database. In this case, there is only one column named StockSymbol 151. This column has all the stock symbols  
30 listed under it.

Next in the Wizard, as shown in FIG. 16, the user defines primary key columns on the table. A table usually has a column or combination of columns whose values uniquely identify each row in the table. This  
5 column (or columns) is called the primary key of the table. The column StockSymbol 161 is selected as the primary key, an added to the lower frame 163 using transfer arrows 162.

Next, an option to filter out unwanted data is  
10 available to the designer to prune the selection of stock symbols, as shown in FIG. 17. For instance, the stock-symbol list had 2000 symbols and the designer was interested only in the symbols starting with 'A', he would have defined a filtering condition based on 'A'  
15 that would have filtered out all other symbols but the ones starting with 'A'. In this example, all input stock symbols are used.

This then completes the definition of the data to be extracted. The group name given to this data i.e.,  
20 ReadStockList 131 (Figure 13), and the column name, i.e., StockSymbol 151 (Figure 15) appears in the starting Data Extraction Wizard form 120 (Figure 12) at GroupName 124 and GroupName 125 for data extraction.

The data read from the Excel file is now understood  
25 by the wizard-based application as a group of data elements organized by a data group named P001\_ReadStockList (see FIG. 19).

The second step is again the Extract Data step, but this time it is Web data that is of interest that must be  
30 extracted from the Web. As shown in FIG. 18 in the Data Extraction Wizard, the designer clicks on the HTML option

182 of the Data Extraction Wizard, and is presented with the options shown.

The designer gives a name 181 to this data extraction (here Initial Navigate to MSN Investor) and thereafter selects the file schema 183 that has the information stored on how to navigate the Web. Also, the designer selects box 184 to specify whether the Web browser will be visible to the user during the actual playback of this application. From the list of data groups already defined, the designer selects a data group 185, and a data element 186 to which the stock symbol from Excel file will be supplied. In FIG. 18, P001\_msnStkInp 185 is the data group, and Symbol 186 is the data element that will receive the stock symbols from the list of symbols read from Excel file (Group - ReadStockList, 124, Element - StockSymbol 125, shown in FIG. 12).

As shown in FIG. 19, the third step, Position Wizard, positions the cursor to the first data element in the data group, named P001\_ReadStockList 191. This is the data group that has the entire list of stock symbols read from the Excel file. The method of moving the cursor is specified in dropdown box 192, and comments added at box 193.

In step 4, in the Assignment Wizard 200 (FIG. 20), the data group that is going to accept a stock symbol, here P001\_msnStkInp.Symbol 201, is forcibly assigned a stock symbol value 202. This step serves as the initialization step. This initializes the Web extraction to go to a well-known Web site, submit SourceValue "IBM" as the stock symbol, and read the result.

Next is Step 5, wherein the Loops Wizard 210 specifies the definition of a conditional loop. The objective is to submit all the stock symbols from the list to the Web page, one at a time, and read the resulting values. So far only the submission-extraction routine (Step 4, FIG. 18) is initialized. Here, a condition is now added that says do the same for rest of the stocks in the list as well. For this, the conditional loop is implemented. The condition 212 in the loop says that until the End of File (EOF) for the data group P001\_ReadStockList (which holds the list of stocks read from the Excel file) is not equal to true, do the steps that follow.

FIG. 22 shows how the condition in FIG. 21 was built. When the designer clicks the Wizard button 214 in the earlier step (FIG. 21), the Condition Builder Wizard 220 is presented. The designer can select any of the application wide data elements, called Global Data Elements' 222, or process specific data elements called Local Data Elements 223. Here, the designer selects P001\_ReadStockList\_EOF data element 224 because this element specifies whether or not the application has finished reading the list of stock symbols. The selection is added to the lower grid 225 by the click of button 226 below the selection. To this element the designer appends the Not Equal statement 227 from the adjacent selection menu 228. At the end the designer adds True 229 manually by typing in the letters. This completes the definition of conditional loop statement.

Thus far, the designer has defined how to read and store the list of stocks from an Excel file



(P001\_ReadStockList), and how to obtain one stock value at a time to submit it to the Web page (P001\_msnStkInp). A link between them has to be established to transfer the data from one to another, and then to submit it to the Web page.

In step 6, as show in Assignment Wizard 230 of FIG. 23, the current stock symbol P001\_ReadStockList.StockSymbol 231 is assigned to the data element P001\_msnStkInp 232 that actually does the job of submitting the stock symbol to the Web page.

In step 7, shown in FIG. 24, the data extraction Wizard defines data extraction from the Web as done previously in step 2. This step is within the conditional loop, and will be repeated until the condition is true. This means that until all the stock symbols are read, this extract data statement will be repeatedly called for each stock symbol read. This is exactly what the designer intended.

Step 8 is another assignment statement, as shown in FIG. 25, which will, in the Assignment Wizard, synchronize the rows in the group that holds the list of stocks, the group that receives the Stock Symbol, and current Stock Value. This is basically a synchronization process whereby the current rows in the two data groups, P001\_ReadStockList 252 and P001\_DataToBeSaved 254 are made to be at the same level. If the stock symbol being read is 10th in the list (P001\_ReadStockList), then the P001\_ReadStockList\_CURRENTROW is 10, and hence the group that stores all the data extracted from the web (P001\_DataToBeSaved) will also be asked to make its current row to 10.

In step 9, the program moves the pointer to the last record in the data group that accepts values from the Web as a response to a submission of one stock symbol to the Web. Thus, as shown in FIG. 26, the data group that  
5 accepts the values from the Web page for one stock symbol, here group P001\_msnStk 262 is made to move its pointer to the last record, by method MoveLast 264. This makes its last row as the current row.

In step 10, the program assigns the extracted value  
10 to the group that will be saved to a file. As shown in FIG. 27, the data that was extracted from the Web, and stored in the data group P001\_msnStk 271 is assigned to another data group, P001\_DataToBeSaved 272. The data group P001\_msnStk 271 is now free to accept Web-extracted  
15 data for a new stock symbol. The data group P001\_dataToBeSaved keeps adding extracted information for all of the stock symbols in the stock list.

In Step 11, shown in FIG. 28, the Position Wizard moves the pointer to next row in the group that holds the  
20 stock symbols. The task of submitting one stock symbol to the Web page, and reading the information back into data variables for one stock symbol ends here. Now to read the next stock symbol and repeat the process for the next stock symbol in list, the designer moves ahead the  
25 pointer one step selecting the method MoveNext 282, so that the next stock symbol in the list becomes the current stock symbol, and is subsequently submitted to the same Web page. This process will continue until all of the stock symbols in the stock list P001\_ReadStockList  
30 281 are read and processed. At the end, the loop condition will not be satisfied, and the application will

come out of the loop, and proceed to the next step in the application.

The next step 12 saves the group to an existing Excel file to save the extracted information to a permanent storage space. As shown in FIG. 29, the designer selects Excel file 291 by clicking the appropriate check box in Save As DataType box 290. The designer also selects the data group 292 that will be supplying information to the Excel file for writing to it. The designer had previously defined a group called P001\_DataToBeSaved that was updated with information for each of the stock symbol in the list. This data group is selected as the supplier of data to be written to a new Excel file.

Step 13 is the design of the IF conditional statement using the Conditional Wizard, shown in FIG. 30. The user was asked in step 1 whether he/she would like to view a final output. That same test is applied here in conjunction with the user's answer. If the user replied with a Yes, then the condition is satisfied, and the statement nested inside this conditional statement is executed. Otherwise, it is not executed.

The last step is the design of the output statement, including headers, footers, and logos. The design of the output works in two parts. The first part is the design of the layout of the page itself. The designer gives a name 311 to this design step (FIG. 31). Then the designer enters the text that will be actually printed on the output and formats the text to his/her desire using various menus 312 available on the Display Wizard.

In the second step of data output design, shown in FIG. 32, the designer decides which data is to be outputted. In this example, the data extracted from the Web was saved in the data group called P001\_DataToBeSaved  
5 321. The same data group is selected to print its element values to the output. The data will be arranged by ascending order based on the company name. The data will be presented in a tabular format. Hence, options such as table border width 324 and border color 325 is  
10 also selected. This completes the entire design process for the application. The final data output is shown in FIG. 33, which displays table 330 in HTML output with the stock details.

The application also provides a Debug phase 342,  
15 accessible from Run menu 341 (FIG. 34). The playback of an application in the Debug phase is to test and correct each individual step in the processes defined. The user can start/stop the business logic after each step for verification (debugging) purposes.

20 To run application in Debug Phase, the designer selects the application file in which the application logic was saved after it was defined in the design mode. This is done through the File and Open menu buttons located on the top of the screen. The program then reads  
25 this file and populates all the information from the file in its memory. The wizard-based tool reproduces the application steps from its memory in the form of a design tree, as had been defined during the design step.

To run the application recently loaded in a debug  
30 mode, the designer goes to the menu option 341 labeled Run. Under this menu, there are submenus like Start and

Debug. The submenu Debug 342 will playback the application in a debug mode one step at a time.

Similarly, if the designer wants to test another process inside this application in isolation, he/she can  
5 do so by selecting the process name from the drop down box 344 near the file menu. This action will refresh the designer window, and load the steps in the designer window that are specific to the selected process. Thereafter, the designer has to go back to the main menu,  
10 select Run, and then select the submenu under it called Start 343. Start has further two submenus called Start Application, and Start Process. The Start Process submenu will start the selected process in debug mode.

To make changes in a process/application step while  
15 in Playback (debug mode) the designer has to double click the step that he/she wants to edit. This will pop up a wizard specific to that step with the relevant current information. The changes made to this wizard will be saved, and a rerun of the process will use the changed  
20 values for that step thereafter. Highlighting a step in the desired process, and clicking the Edit button 345 on the tool bar can also perform the edit operation. The click on Edit menu 345 presents three new submenus Application, Current Selection, and Delete Current Node.  
25 The Current Selection submenu allows the designer to edit the contents of the highlighted step.

Once the playback starts, the wizard-based programming application reads each step of the process tree one by one. The verb at each step helps the  
30 application to decide what kind of action is to be performed next. The conditional loops appear in the

design steps only once. The application development software preferably intelligently handles such conditions by actually looping the steps until the looping condition is satisfied.

5 After being satisfied that the logic is working correctly, the user can schedule the business logic to run automatically at pre-determined frequencies, including on demand. In the run phase, the logic runs automatically without any user intervention. In this  
10 mode, the user does not see the design steps, nor does he see the wizards. The job of selecting processes and submitting them to the application is performed through the Scheduler. This sometimes also is referred to as the silent mode of operation.

15 In the run phase, the application reads the application file in its memory, and arranges the steps of the processes internally in its memory and does not display them in a tree format as is done in the design phase.

20 A schedule designer and a scheduler are used in conjunction to create a schedule of processes and then run them automatically on the defined schedule. The schedule designer, shown in FIG. 35, can add, update and delete processes to a schedule file. It can also create a  
25 brand new scheduler file and add processes to it. The Open button 350 opens an existing or new scheduler file. If there are already some processes in the existing scheduler file, they are listed in the Scheduled Test grid 351. A new process can be added by clicking the "+"  
30 (plus) button 352. This brings up another form that reads the InstaKnow design files and picks up the

processes, and arranges them in a grid. The designer can selectively pick processes from this wizard form to transfer them to the actual list of scheduled processes.

To delete a process from the grid and from the  
5 scheduler, highlight the process in the Scheduled Tests grid 351 and clicks the "-" button 353 to delete it.

To update a process, the user highlights the process in the Scheduled Tests grid 351 then selects one of the interval options, namely Daily, Hourly, or ASAP (for As  
10 Soon As Possible) in Interval column 354. If the process is ASAP, it is assigned a priority code of zero (0) in Priority column 355 and goes as the topmost priority. Then, the users gives the actual time that the process in concern is supposed to run in Column 356. For example,  
15 if the user wants to run a process P1 every hour at 35 minutes past the hour, the user will check the 'Hourly' option, and type 35:00 in the adjacent box. After the information is provided, the user clicks the "=" (equal) button 357, and the information will be updated in the  
20 highlighted row of the grid in their appropriate places.

After all the processes have been assigned a time to start and priority, the scheduler file should be saved using Save button 358 to save the updated information to be used in the InstaKnow Scheduler, shown in FIG. 36.

25 A sort option is provided to the designer by clicking on Sort Grid(s) button 359 to sort the Scheduled Tests grid, and the Historical Log of Processes grids. The sorting operation is performed based on the grid columns such as priority, process ID, and application ID.

The lower grid Historical Log of Processes 360, displays the processes that have been already processed by the Scheduler. This allows the designer to perform checks on the already run processes for results, and errors.

The Scheduler (FIG. 36) is in charge of actually submitting the scheduled processes to the application. The scheduler file that was created is opened to read the scheduler information. The Show in Grid button 361 actually displays the eligible processes in the 'Processes to be Submitted' grid 362 below. The Historical Log File 363 also is opened so that the processed processes can be written out to the historical log file with appropriate messages.

A 'Time interval' and the 'Time to start' are assigned. These two times work in conjunction to decide which process will be pulled, and be submitted to InstaKnow application, and which process actually is submitted. The time interval is the time between the start time and the future time between which all the processes will be scheduled. In FIG. 36 the time interval 363 is 60 minutes, and the start time 364 is 11/12/1999 at 11:15:00 am. When the scheduler file is opened, the InstaKnow scheduler checks the 'time to be run' time stamps on each process, and decides if that process falls between 11:15:00 am and 60 minutes beyond it, i.e., 12:15:00. FIG. 36 shows that all the three processes have been scheduled hourly to be run ranging from 45 minutes to 55 minutes, and three fall between the time interval specified. Therefore, they will be scheduled to run.



A click on the Run button 365 actually starts the scheduling process. The first process in the Processes To be Submitted grid 362 is removed from this grid and brought in the lower grid 366 named 'Active Processes' to show which process has been scheduled.

Once this process is finished running, it is removed from the Active Process grid 366 and is replaced by the new active process. Before a newly active process replaces the current process, the information is written out to the historical log file. The cycle continues until all the processes from the Processes to be Submitted grid 362 have been submitted.

After a complete cycle, the Scheduler goes in the sleep mode till the time interval is completed. In the example, Scheduler will wake up every one-hour, schedule the processes from the scheduler file, and start submitting them to the InstaKnow application.

Using an improved feature of transportable intelligence, multiple Web based computing resources cannot only automatically share data with each other, they can automatically share business rules or business intelligence with each other when required. The business intelligence can be immediately executed by the receiving computer.

The application development program keeps all conditional business logic as a data file/string called a knowledge Element or a Knowlet. This business intelligence can be supplied by one computer to other computers by simply transferring the Knowlet string over the Internet to an authorized and willing computer, which

can then immediately execute that business logic under its own control.

In one business example of transportable business intelligence, a user can determine the best shipping cost  
5 quote using custom criteria. A customer visiting a Web-based Shipping Marketplace requests a quote for a commodity purchase. The complete quote consists of the commodity price plus the shipping price. However, in this case, the particular customer has the unique demand  
10 to find the cheapest shipper who will give at least a 15% discount on shipping charges if the shipping charges exceed 3% of the purchase price.

Since this conditional business rule is non-standard, it cannot be pre-supported by the Commodity or  
15 Shipping Marketplace. One of the few ways to support such custom business rules is to actually transfer the unique business rules, not just context sensitive business data, to the collaborating party (in this example the Shipping Marketplace) by encapsulating the  
20 rules as Knowlet data string and transferring it over the Internet to a partner.

Knowlets can be forwarded from partner to partner-unlimited number of times. Additional custom data and intelligence can be added to Knowlets in context specific  
25 ways before they are forwarded.

Using the application development software's 'transportable intelligence' capability, innovative services can be provided by collaborating partners across the Internet to satisfy customized demands and add value  
30 to the value chain.

As these and other variations and combinations of the features discussed above can be utilized without departing from the present invention as defined by the claims, the foregoing description of the preferred  
 5 embodiments should be taken by way of illustration rather than by way of limitation of the present invention.

WHAT IS CLAIMED IS:

1. A computer-implemented method for running applications on a server computer and generating Web page information to be displayed on one or more client computers connected to the server computer via the Internet, comprising:

- 10 (a) sending data from a client computer to a server computer running a DLL, said data comprising an identify of a user selected application, Web page information associated with a Web page displayed on said client computer, and user-entered information used with said application;
- 15 (b) running said DLL to retrieve said data and to identify one or more executable processes within said selected application;
- (c) executing said identified processes within said DLL in association with said user-information to generate output information;
- 20 (d) generating Web page information used to form a Web page viewable at said client computer by said DLL, said Web page information containing said output information; and
- (e) forwarding said Web page information to said client computer for display.

25 2. A method as claimed in claim 1, wherein said forwarding comprises forwarding said Web page information to an ASP layer.

3. A method as claimed in claim 1, wherein said Web page information comprises HTML output.

4. A method as claimed in claim 1, wherein said Web page information comprises HTML tags.

5. A method as claimed in claim 1, wherein said Web page information comprises an incoming screen name  
5 from the current screen being viewed at said client computer.

6. A method as claimed in claim 1, wherein said DLL runs a routing subroutine to select said one or more executable processes by reference to parameters of  
10 incoming screen name, application file name and a selected screen element.

7. A method as claimed in claim 1, wherein said client computer is configured to run a Web browser for sending and receiving information to and from said server  
15 computer.

8. A computer-implemented method for running applications on a server computer connected via a network to one or more client computers, comprising:

- 20 (a) receiving requests at said server computer from a client computer running a browser program;
- (b) running an application in response to said requests;
- (c) constructing coded information used to form output by said browser program based on results generated from said application; and  
25
- (d) forwarding said coded information to said browser program, wherein all necessary business logic for receiving said requests, running said application, and constructing said coded

information is contained within a single linkable library of executable functions.

9. A method as claimed in claim 8, wherein said single linkable library of executable functions resides  
5 exclusively in said Web server computer.

10. A method as claimed in claim 9, wherein said single linkable library of executable functions comprises a DLL.

11. A computer-implemented method for running  
10 applications in a networked computing environment comprising:

- (a) sending requests from a client computer running a browser program;
- (b) receiving said requests at a server computer;
- 15 (c) running an application in response to said requests;
- (d) constructing output information at said server computer in a format used by said browser program based on results generated from said application; and
- 20 (e) forwarding said output information to said browser program for display at said client computer, wherein all necessary business logic for receiving said requests, running said application, and constructing said output information is contained within a single linkable library of executable functions.
- 25

12. A method as claimed in claim 11, wherein said application is run at least in part on server computer.

13. A computer-based system for running applications in a networked computing environment comprising:

- (a) a client computer running a browser program;
- 5 (b) a server computer for receiving requests from said client computer over a network connecting said client and server computers; and
- (c) said server computer running an application to respond to said requests, construct output  
10 information in a format usable by said browser program based on results generated from said application, and forward said output information to said browser program for display at said client computer, wherein all necessary  
15 business logic for receiving said requests, running said application, and constructing said output information is contained within a single linkable library of executable functions.

14. A computer readable medium for storing a set of  
20 instructions for controlling a server computer in a networked computing environment including a client computer running a browser program, a server computer for receiving requests from said client computer, and a computer network interfacing said client and server  
25 computers, said medium comprising:

- (a) a single linkable library of executable functions; and
- (b) set of instructions resident in server computer for causing said server computer to (i) run an  
30 application to respond to requests from said

client computer, (ii) construct output information in a format used by said browser program based on results generated from said application, and (iii) forward said output information to said browser program for display at said client computer, wherein all necessary business logic for receiving said requests, running said application, and constructing said output information is contained within said single linkable library of executable functions.

15. A computer-implemented method for developing a business rules based application, comprising:

- (a) providing a set of wizards selectable by a user;
- (b) providing a set of verbs selectable by the user to implement business rules used in said application;
- (c) choosing at least one of said verbs to define an action to be performed by said application when executed; and
- (d) selecting at least one of said wizards to specify actions to be performed by said application when executed.

16. A computer-implemented method for developing a business rules based application, comprising:

- (e) providing a set of wizards selectable by a user, including a data extraction wizard and an assignment wizard;



- (f) providing a set of verbs to be used to implement business rules used in said application;
- 5 (g) choosing a verb to define at least one input criteria to be specified by the user of said application when executed;
- (h) selecting said data extraction wizard to instruct said application to extract initial data from a first location;
- 10 (i) selecting said assignment wizard to specify a location for said application to store said initial data;
- (j) selecting said data extraction wizard to instruct said application to extract secondary data from a second location, said secondary data being selected based on said initial data; and
- 15 (k) selecting said assignment wizard to specify a location for said application to store said secondary data.
- 20

#### ABSTRACT OF THE DISCLOSURE

A computer-implemented method and system for developing and running applications in a networked computing environment includes one aspect in which  
5 requests are sent from a client computer running a browser program and receiving the requests at a server computer. The server computer runs an application in response to the requests and constructs output information at the server computer in a format usable by  
10 the browser program, based on results generated from the application. The server computer forwards the output information to the browser program for display at the client computer. All of the necessary business logic for receiving the requests, running the application, and  
15 constructing the output information is contained within a single linkable library of executable functions. Another aspect provides for the development of customized application using business rules and logic with the need for a programmer.

20 278487\_1

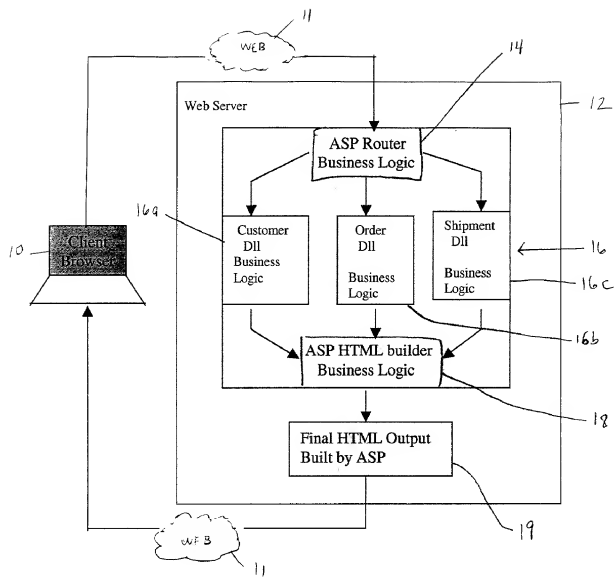


FIG. 1

(continued)

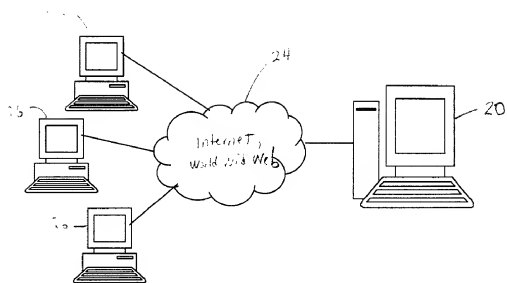


FIG. 2

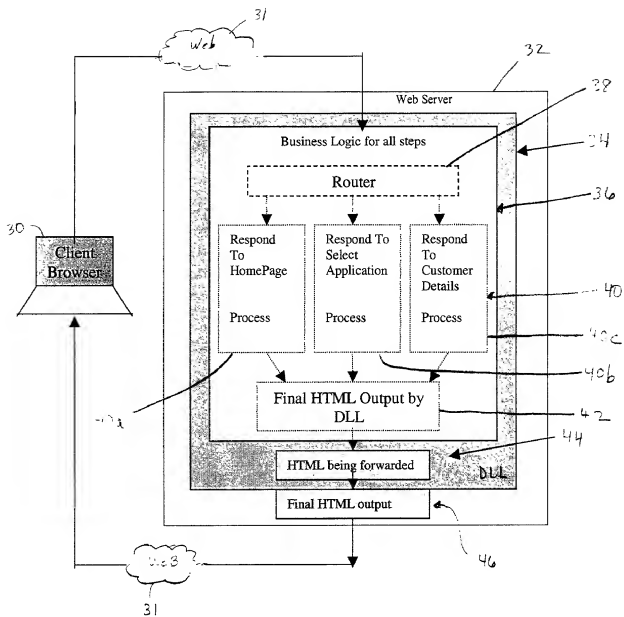


FIG 3

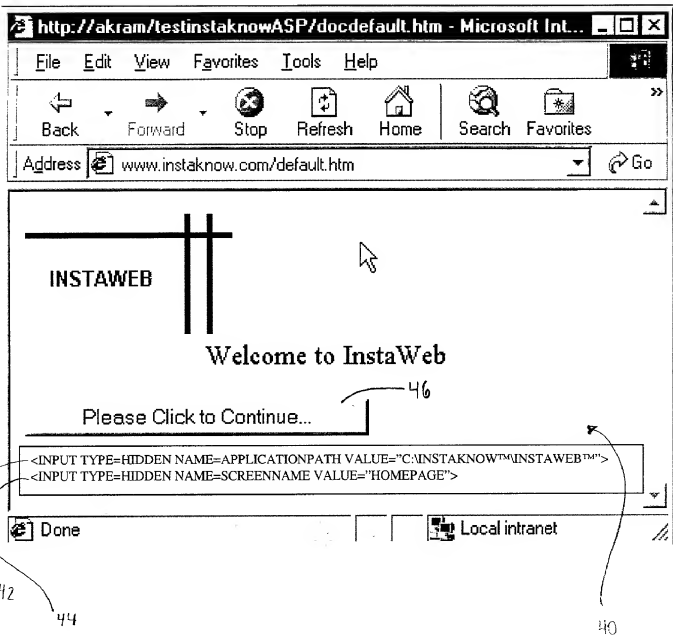


FIG 4

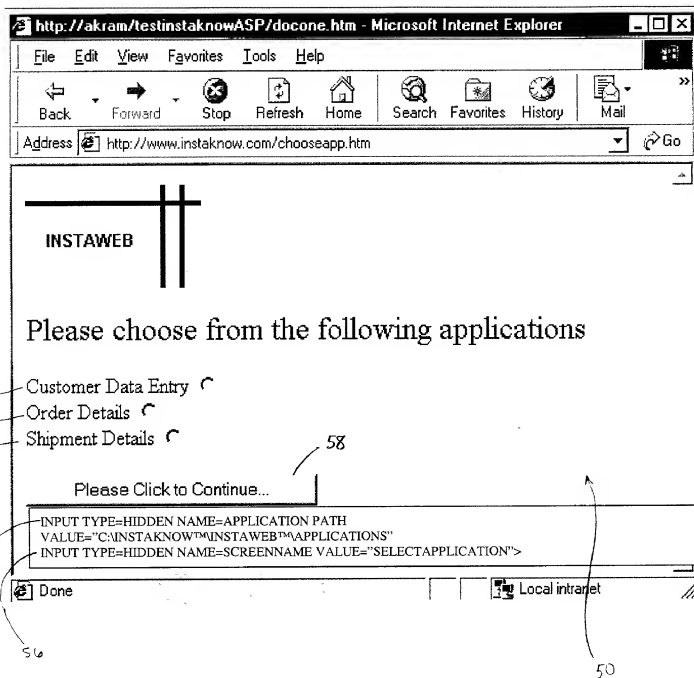


FIG. 5

http://akram/testinstaknowASP/doctwo.htm - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail

Address www.instaknow.com/customerdetails.htm Go

INSTAWEB

Please enter the following information

Name  City

Date of Birth  E-mail Address

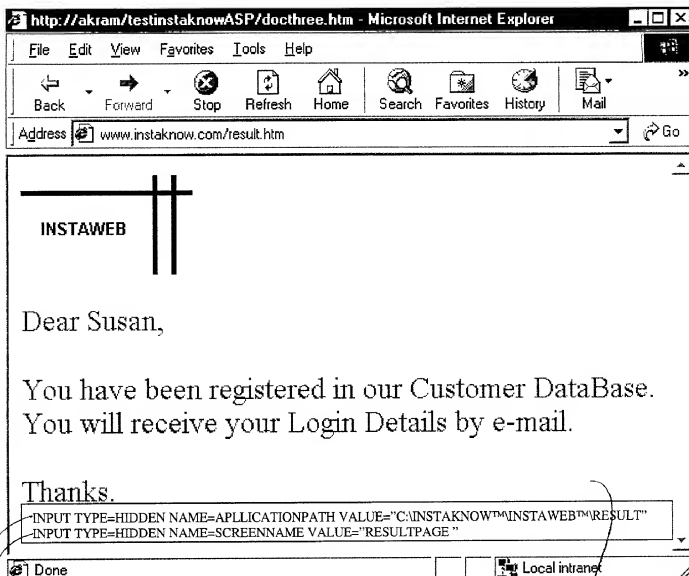
Please Click to Continue...

INPUT TYPE=HIDDEN NAME=APPLICATIONPATH  
VALUE="C:\INSTAWEB\TM\INSTAKNOW\TM\CUSTOMER">

Done Local intranet

FIG. 6





277813 1.DOC

76

74

F16.7

70

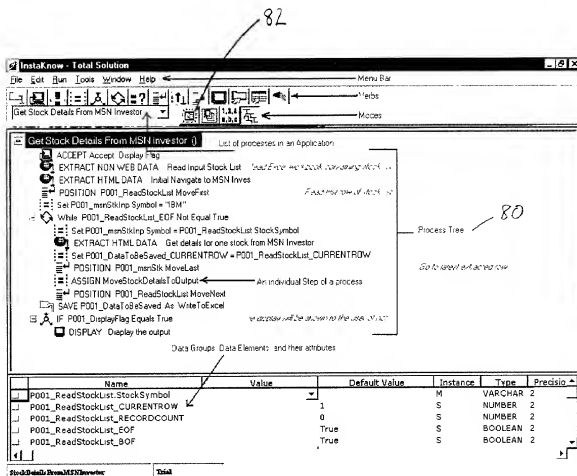


FIG 8

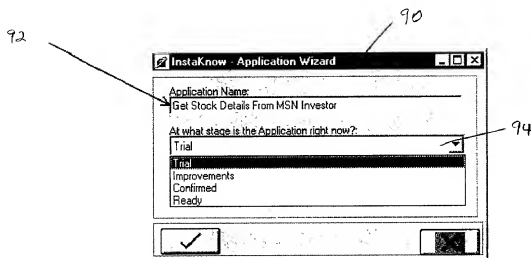


FIG. 9

**InstaKnow - Process Wizard**

Name:  102

Description:

At what stage is the Application right now?:  100

At what stage is the Application right now?:  104

Color Scheme:  106

Language:  108

☐ Audio Enabled 109

☒

FIG. 10

**InstaKnow - Accept** 116

Name:  116

Comments:

Select the Group:  111

Which Data Elements Do you want the User to Enter at RunTime

	DataElements	DataTypes	DefaultValue	
14	P001_DataToBeSaved_RECORDCOUNT	Integer	0	This variable store
15	P001_DataToBeSaved_EOF	Boolean	True	This variable store
16	P001_DataToBeSaved_EOF	Boolean	True	This variable store
17	P001_DisplayFlag	Boolean	True	Decides whether to
18	P001_DisplayFlag_SupInfo	VarChar		Decides whether to

110

112

	DataElements	Prompt	Marking	E
1	P001_DisplayFlag	Do you wish to see the output display ? (Yes/No)		

113

115

☒

☐ Only to current statement  
☒ Next to current statement

114

FIG. 11

InstaKnow - Data Extraction Wizard

Name: Read Input Stock List

Comments: Read Excel workbook containing stock list

☒ HTML
 ☐ XML
 ☐ Other Data sources

Show Interface Details

GroupName	ColumnName
ReadStockList	StockSymbol

☒ Child to current statement  
☐ Next to current statement

FIG. 12

InstaKnow - GET RELATIONAL DATA

Data Source

Group Name

ReadStockList

Enter the details of your data source.

Type

MS Excel

Path / URL:

C:\TotalSolution\App1\InputStockList.xls

Rows to Extract

999

Zero by default gets all rows

☐ Disconnect after Getting Data

Cursor Type

☐ Static

☐ Dynamic

☐ Key

☒ Forward only

Finish

Previous

Next

Preview

Cancel

FIG. 13

## Source Tables

Select the tables to use from your data source.

Sheet1\$Database

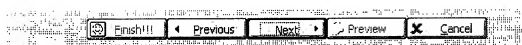


FIG. 14

## Column definition

Define columns you want to use in the Business Logic

StockSymbol

Group By:

Select the Key Columns in the next step first and then come back here to Define Group By columns

Finish!!! Previous Next Preview Cancel

FIG. 15

InstaKnow - GET RELATIONAL DATA

Key columns

Key columns identify unique rows.

StockSymbol



StockSymbol

ASC

☐ Sort before compare

To Define a Group By column go to the previous step

Finish!!! Previous Next Preview Cancel

F16.16



Filtering

Define filter conditions, if any to narrow down the data



Finish!!! Previous Next Preview Cancel

FIG 17

**InstaKnow - Data Extraction Wizard**

Name:  Comments:

☒ HTML ☐ XML ☐ Other Data sources

Select the schema:   Time Out (sec)   
☒ Load schema from files ☐ Load schema that is supplied ☐ Start from current page  
☒ Show Browser at run time

Select the Group:  Starting URL:  Select the Variable:

GroupName	Column Name
msnStkInp	Symbol Var Char

☒ Child to current statement ☒ Next to current statement

FIG 18

**InstaKnow - Position Wizard**

Select a group:

Select a method:

Comments:

☒ Child to current statement ☒ Next to current statement

FIG. 19

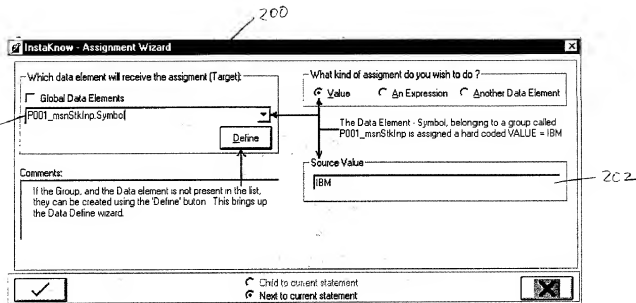


FIG. 20

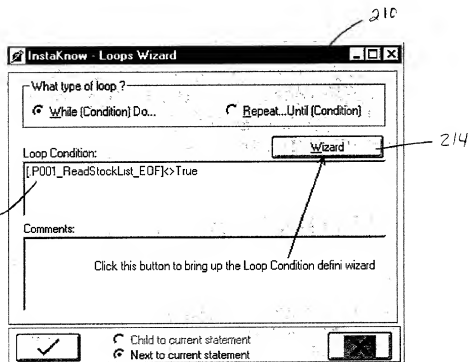
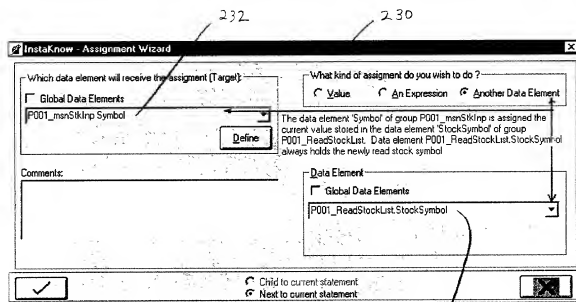
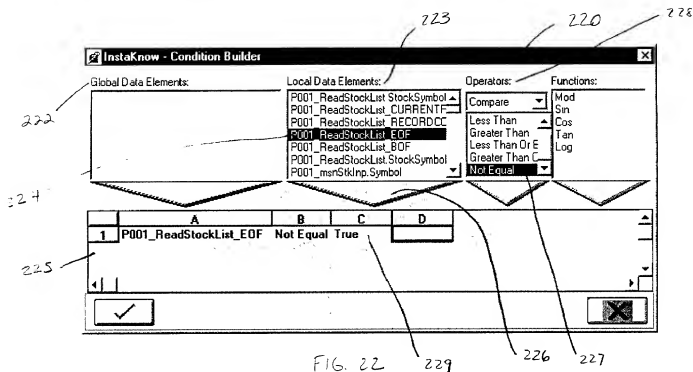


FIG. 21



**InstaKnow - Data Extraction Wizard**

Name:  Comments:

☒ HTML ☐ XML ☐ Other Data sources

Select the schema:  Time Out (sec)

☒ Load schema from files ☐ Load schema that is supplied

☐ Start from current page ☒ Show Browser at run time

Select the Group:  Starting URL:  Select the Variable:

GroupName	ColumnName
minStock	Symbol Var Char

☒ Child to current statement ☐ Next to current statement

FIG 24

**InstaKnow - Assignment Wizard**

Which data element will receive the assignment (Target):

☒ Global Data Elements

Comments:  
If the Group, and the Data Element is not present in the list, it can be created using the 'Define' button. This brings up the Define Data wizard

What kind of assignment do you wish to do?

☐ Value ☐ An Expression ☒ Another Data Element

The data variable - CurrentRow in Group P001\_DataToBeSaved is assigned the current value stored in another data variable - Currentrow of the P001\_ReadStockList group, which always holds the current value of the Stock for the corresponding Stock symbol


Data Element


☒ Global Data Elements

☒ Child to current statement ☐ Next to current statement

FIG 25

**InstaKnow - Position Wizard**

Select a group  
 

Select a method  
 

Comments:

☒ Child to current statement  
☐ Next to current statement

262


264


FIG 26

**InstaKnow - Assignment wizard**

Name of the Assignment:

Enter Comments:

Select the Receiving Group:  

Select the Sending Group:  

COLUMN NAME	TYPE	TRANSFORMATION
P001_DataToBeSaved.CompanyName	VarChar	P001_msnStk.CompanyName
P001_DataToBeSaved.CompanyName_SupInfo	BinaryString	P001_msnStk.CompanyName_SupInfo
P001_DataToBeSaved.Last	VarChar	P001_msnStk.Last
P001_DataToBeSaved.Last_SupInfo	BinaryString	P001_msnStk.Last_SupInfo
P001_DataToBeSaved.Open	VarChar	P001_msnStk.Open
P001_DataToBeSaved.Open_SupInfo	BinaryString	P001_msnStk.Open_SupInfo
P001_DataToBeSaved.Change	VarChar	P001_msnStk.Change
P001_DataToBeSaved.Change_SupInfo	BinaryString	P001_msnStk.Change_SupInfo
P001_DataToBeSaved.PreviousClose	VarChar	P001_msnStk.PreviousClose
P001_DataToBeSaved.PreviousClose_SupInfo	BinaryString	P001_msnStk.PreviousClose_SupInfo

☐ Find closest col name match

☒ Child to current statement  
☐ Next to current statement

27

FIG 27

**InstaKnow - Position Wizard**

Select a group

Select a method

Comments:

☒ Child to current statement  
☒ Next to current statement

Fig 28

**InstaKnow - Save As Wizard**

Save As Name:

Comments:  
 The extracted stock values stored in this group will be written to the excel file against the corresponding stock symbols.

Select the Group to be saved

☐ Check to include the Supporting Information column  
 If there are spaces in the Recordset Field Names, replace spaces with  
☐ Hyphen ☒ Underscore

Save as Data Type  
☐ XML  
☒ Excel  
☐ Flat File

Number of Rows to Extract in Output

☒ Child to current statement  
☒ Next to current statement

Fig. 29

**InstaKnow - Condition Wizard**

☐ Do you wish to take some action if this condition is NOT satisfied

Comments:  
This IF statement will decide whether the display will be shown to the user or not.

Left Hand Side Condition: Wizard

[P001\_DisplayFlag]=True

☒ Child to current statement  
☐ Next to current statement

FIG. 30

**InstaKnow - Display**

**Name**  
Display the output

**Comments**

**Enter New Section**  
Result of Process run to get stock details  
Do  
=

Double Click on Item for Section Details

**Enter Header Info**

**Color And Font** **Header** **Footer**

Press F1 for Preview

Stocks Details

Double Click on Item to set Details

Select Alignment  
CENTER

Top Logo ☒  
Bottom Logo ☐

Preview ☐ Show browser at run time

☒ Child to current statement  
☐ Next to current statement

FIG. 31



**Details of Result of Process run to get stock details**

Select Data Group: P001\_DataToBeSaved Row Level Highlight Routine: [ ]

Select Colors and Font

ColorAndFont: [ ] Header: [ ] Footer: [ ]

Press F1 for Preview

Stock Details: [ ]

Select Alignment: CENTER

Group Details

GroupElements: [ ] Data Type: [ ]

+ - ++ --

	ColumnName	ColumnHead	HeaderFbf	ColumnFbf	Break	Asc/Desc	LineType	VisibleDescr	Ac
1	P001_DataTo	CompanyNam			┘	┘			
2	P001_DataTo Last				┘	┘			
3	P001_DataTo Open				┘	┘			
4	P001_DataTo Change				┘	┘			

Line Type for Details: [ ] Border Width: 2

Grand Totals: [ ] Grid lines: [ ] Alignment: CENTER

Border Color: [ ] Table Width: [ ]

[ ] [ ] Collect

Fig 32

324


325

InstaKnow Output - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Refresh Home Search Favorites History Mail Print Edit

Address [C:\TotalSolution\StockDetails\fromMSNInvestor\Get Stock Details From MSN Investor\Display the output.htm](http://C:\TotalSolution\StockDetails\fromMSNInvestor\Get Stock Details From MSN Investor\Display the output.htm) Go Links

 InstaKnow

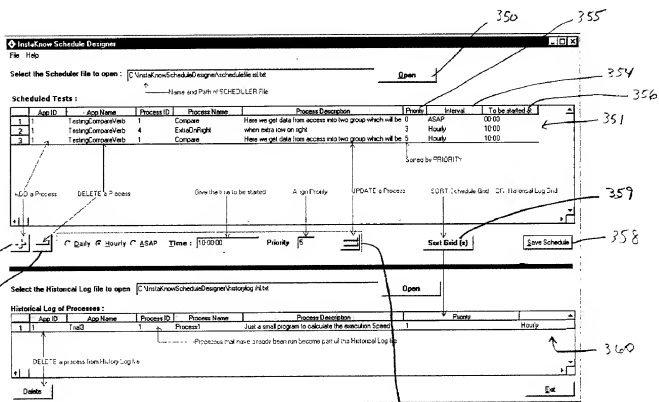
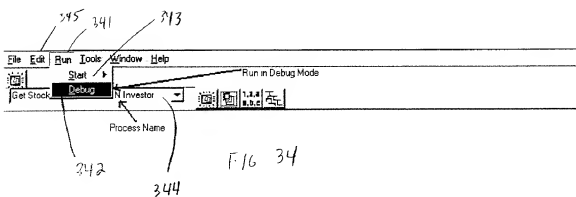
*InstaKnow Output*

Stock Details

CompanyName	Last	Open	Change	PctChange	Volume	PE	EarningsPerShare	DivPerShare
Dell Computer Corporation	41 11/16	40 7/8	+7/8	+2.14%	26.55 M	57.70	0.62	NA
Microsoft Corporation	87 5/16	86 15/16	+5/16	+0.36%	29.42 M	57.30	1.62	NA
International Business Machines Corporation	94 5/8	94 5/8	+9/16	+0.60%	8.537 M	22.30	4.22	0.48
Compaq Computer Corporation	21 3/4	21 11/16	+3/16	+0.87%	13.08 M	33.70	0.64	0.08

Done My Computer

F16.33



**Frozen-InstaKnow Scheduler**

Scheduler file created in InstaKnow Schedule Designer

Scheduler File Name : C:\InstaKnow\ScheduleDesigner\schedulerfile.txt Browse

History Log File name : C:\InstaKnow\ScheduleDesigner\Instalogfile.txt Browse

Select an interval : 60 minutes Time Interval Time to Start 11/12/1999 11:15 am

Processes to be Submitted:

363

364

361

Show in Grid Run 365 Resume Delete

App	App Name	Process	Process Name	Process Description	Priority	Interval	To be started at
1	TestingCompareVer 1	Compare	Compare	Here we get data from access 2	2	Hourly	45:00
2	TestingCompareVer 4	ExtraOnFlight	ExtraOnFlight	when extra row on night	3	Hourly	55:00

Processes scheduled to run in order of priority. These processes were picked up from InstaKnow scheduler file. Scheduled time to run. These processes are waiting to be executed.

362

Active Process:

Process Running right Now

App	App Name	Process	Process Name	Process Description	Priority	Interval	To be started at
1	TestingCompareVer 1	Compare	Compare	Here we get data from ac1	Hourly	50:00	

366

Show Web Browser Close

Frozen 1:43:15 PM Number of processes scheduled:

277948\_1.DOC

F 16. 36

**DECLARATION FOR UTILITY OR DESIGN PATENT APPLICATION****ATTORNEY'S DOCKET NO.: INSTAK 3.0-001**

As a below-named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name;

I believe I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled:

**METHOD AND SYSTEM OF DEPLOYING SERVER-BASED APPLICATIONS** the specification of which☒ is attached hereto☐ was filed on \_\_\_\_\_ as United States Application Number or PCT International Application Number \_\_\_\_\_ and was amended on \_\_\_\_\_ (if applicable).

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment specifically referred to above.

I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, § 119(a)-(4) of any foreign application(s) for patent or inventor's certificate or § 365(a) of any PCT international application which designated at least one country other than the United States of America, listed below and have also identified below any foreign application for patent or inventor's certificate, or any PCT international application having a filing date before that of the application on which priority is claimed:

PRIOR FOREIGN APPLICATION(S)			
COUNTRY	APPLICATION NUMBER	DATE OF FILING (month, day, year)	PRIORITY CLAIMED
			YES <input type="checkbox"/> NO <input type="checkbox"/>
			YES <input type="checkbox"/> NO <input type="checkbox"/>
			YES <input type="checkbox"/> NO <input type="checkbox"/>

LISTING OF FOREIGN APPLICATIONS CONTINUED ON PAGE 3 HEREOF ☐ YES ☒ NO

I hereby claim the benefit under Title 35, United States Code, § 119(e) of any United States provisional application(s) listed below:

Application Number: 60/174,747

Filing Date: January 4, 2000

Application Number: 60/166,247

Filing Date: November 18, 1999

Application Number: 60/171,143

Filing Date: December 16, 1999

I hereby claim the benefit under Title 35, United States Code, § 120 of any United States application(s), or § 365(c) of any PCT international application designating the United States of America, listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT international application in the manner provided by the first paragraph of Title 35, United States Code, § 112, I acknowledge the duty to disclose information which is material to patentability as defined in Title 37, Code of Federal Regulations, § 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application:

U.S. Parent Application Serial Number:

Parent Filing Date:

Parent Patent No.:

U.S. Parent Application Serial Number:

Parent Filing Date:

Parent Patent No.:

PCT Parent Number:

Parent Filing Date:

LISTING OF US APPLICATIONS CONTINUED ON PAGE 3 HEREOF: ☐ YES ☒ NO

**POWER OF ATTORNEY:** As a named inventor, I hereby appoint the following registered practitioner(s) to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith: Customer Number 000530

**DIRECT ALL CORRESPONDENCE TO:** Customer No. 000530

**DECLARATION -- Page 2**

ATTORNEY DOCKET NO. INSTAK 3.0-001

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further, that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full name of sole or first inventor (given name, family name): PRAMOD KHANDEKAR

Inventor's signature P. S. Khandekar Date 11/15/2000

Residence: Edison, New Jersey Citizenship: USA

Post Office Address: 1624 Raspberry Court, Edison, New Jersey 08817

Full name of second joint inventor, if any (given name, family name)

Second Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of third joint inventor, if any (given name, family name):

Third Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of fourth joint inventor, if any (given name, family name):

Fourth Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of fifth joint inventor (given name, family name):

Fifth Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of sixth joint inventor, if any (given name, family name):

Sixth Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of seventh joint inventor, if any (given name, family name):

Seventh Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_

Full name of eighth joint inventor, if any (given name, family name):

Eighth Inventor's signature \_\_\_\_\_ Date \_\_\_\_\_

Residence: \_\_\_\_\_ Citizenship: \_\_\_\_\_

Post Office Address: \_\_\_\_\_